Einführung in die Programmierung

WS 2019/2020, Blatt 03

Bauhaus-Universität Weimar

Prof. Norbert Siegmund
André Karge

Ausgabe: 18.11.2019
Abgabetermin: Montag, 25.11.2019, 11:00
Besprechung: 28.11.2019

Bitte lesen Sie die folgenden Informationen zum Übungsablauf **sorgfältig** durch.

Grundsätzlich – wenn nicht anders angegeben – sind die Lösungen zu den Übungen zu Einführung in die Programmierung jeden **Montag bis spätestens 11:00 Uhr** an André Karge per E-Mail zu schicken.

Schreiben Sie bitte im Betreff Ihrer E-Mail Ihre **Teamnummer** sowie die Nummer des Übungsblattes. In der E-Mail schreiben Sie bitte zusätzlich Ihren **Namen** und **Matrikelnummer**. Die Lösungen für Sie bitte als Java Dateien als Anlage hinzu. Es werden **keine** kompilierten Dateien, wie *.class oder *.jar angenommen.

Übungen müssen von **minimal zwei** und **maximal drei** Studierenden aus derselben Übungsgruppe in einem festen Team bearbeitet werden (Ausnahmen nur auf Anfrage beim Übungsleiter). Pro Team soll die Lösung nur einmal abgegeben werden. Aufgaben sollen **im Team gelöst** und nicht nur vom Team abgegeben werden. Sie müssen mindestens **50%** dieser Punkte für eine Zulassung zur Prüfung erreichen. Das **Abschreiben** identischer Lösungen wird jeweils mit 0 Punkten bewertet.

Bei Fragen oder Unklarheiten wenden Sie sich bitte **vor der Abgabe** des Übungsblattes an den Übungsleiter (per E-Mail oder persönlich). Es soll nie jemand sagen müssen: „Wir haben die Aufgabe nicht verstanden und konnten sie daher nicht bearbeiten."
Weitere Informationen, wie aktuelle Ankündigungen, finden Sie online (`https://www.uni-weimar.de/de/medien/professuren/intelligente-softwaresysteme/lehre/`) unter Einführung in die Programmierung

Aufgabe 1 Battleship (8 Punkte)

Schreiben Sie ein Programm, das ein "Ein-Schuß-Schiffeversenken" realisiert. Legen Sie hierzu in ihrem Programmcode ein festes zweidimensionales Array der Größe 10×10 an, das für jede Koordinate (x, y) speichert, ob sich an dieser Stelle ein Schiff(teil) befindet. Überlegen sie sich, welcher Datentyp hierfür am Besten geeignet ist. Als Beispiel können Sie folgende Seekarte verwenden:

						o		o	
	o	o	o			o		o	
								o	
					o	o		o	
o	o								
				o	o	o	o	o	
	o		o	o	o	o			o
	o								o
	o			o	o				o

Fragen Sie den Benutzer nach einer x und y Koordinate. Geben Sie dann aus, ob sich an der entsprechenden Stelle ein Schiff(teil) befindet. Prüfen Sie auch ab, ob die angegebene Koordinate zulässig ist.

Aufgabe 2 Static (2+3+2+1 Punkte)

Erweitern Sie die bekannte Klasse `Triangle` (siehe material03.zip). Verwenden Sie dabei `static` wo notwendig.

(a) Jedes Dreieck soll wissen, wieviele Dreiecke generiert worden sind. Erstellen Sie eine Methode um diese Zahl auszulesen.

(b) Jedes Dreieck soll wissen, welche minimale rechteckige Fläche notewendig ist um alle generierten Dreiecke darauf zu zeichnen.

(c) Jedes Dreieck hat eine Methode um die Komplementärfläche (zur Fläche aus (b)) zu berechnen.

(d) Erstellen Sie eine eigene Klasse um Ihre Implementierung zu testen.

Aufgabe 3 Arrays (5 Punkte)

Schreiben Sie ein Java Programm, welches die Zahlen 1 bis 7 im Array
`int[] arrayToCount =` $\{2, 4, 1, 2, 15, 123, 2, 3, 5, 6, 5, 5, 7, 7, 1, 1, 2, 2, 3\}$; zählt. Speichern Sie die momentane Anzahl der jeweiligen Ziffern in einem Integer Array ab. Die Zahl 1 soll dabei im ersten Slot stehen, die Zahl 7 im letzten Slot (die $i$te Zahl soll also an $(i-1)$ter Array Stelle stehen). Sollte eine Zahl im `arrayToCount` vorkommen, die nicht im Interval $[1; 7]$ liegt, so ist dies auf der Konsole auszugeben. Verwenden Sie zur Überprüfung und zum Zählen eine Switchblock.